Haaga-Helia

Abstract

Haaga-Helia
University of Applied Sciences

Path of an object to a game engine via 3D reconstruction

Vincent Schülé

Bachelor's Thesis

Degree Program in BIT

2016

Author	**Year of entry**
Vincent Schülé	

Haaga-Helia

Abstract

	2016
Title of report Path of an object to a game engine via 3D reconstruction	**Number of report pages and attachment pages** 41+8
Advisor Heikki Hietala	

With the advancement in technology and upgrade in pc available resources, videogames assets become more and more realist and detailed. 3D reconstruction can be used to facilitate the design of realistic object.

The purpose of this study was to recreate an object digitally and use it in a videogame. Additionally, the reader should be able to follow this document to create their own project. The thesis does not cover the conception of a new tool, however it uses interesting software and compare them.

The thesis was implemented by using the same photoset to compare the reconstruction of two different software. Afterwards, the best result was optimized with both a manual and an automatic solution. The optimized output was then exported in a game engine.

The result concluded in two usable assets in a prototype videogame. Demonstrating the accessibility of 3D reconstruction for this industry.

Keywords
3D reconstruction, photogrammetry, game engine, 3D scan

Table of contents

Table of figures

1 Introduction

The constant advancement in technologies means that computer can render 3D models with more and more details. But getting more accurate also means exponential work for 3D artist. On the other side, base quality of textures and shaders can create nice graphic without too much thinking from the developer. Problem being, the result will often appear fake or unrealistic to the audience, even if good looking. The brain seems to register problems and understands when something is not normal. (Poznanski, 2014)

Enter 3D reconstruction and photogrammetry, these technologies can use laser scan, pictures or even frames of a video to recreate 3D digital models of a real object, with both the shape and the texture. The result is a photorealistic representation of the target. However, the reconstruction generates resource costly models. For videogames, the mesh needs to be optimized into a less detailed hull while keeping the realistic features. Plus, the goal of the 3D reconstruction is to get a coherent object, but the photorealistic texture can be edited into a more stylish representation depending on the game needs.

This thesis examines 3D technologies, reconstruction and how to import real elements into videogames. It notes the challenges of using photogrammetry as a developer or designer, as well as approaches to clear these problems. These recommendations are targeted to 3D artists and game developer. It delivers some explanations for basic 3D terminology but interest in the subject would help minor misunderstandings.

The research presents photogrammetry as a solution for modelling realist assets in videogames. For this, I rely on my personal experiences of 3D design, sources and the use of photogrammetry in already released project like *The Vanishing of Ethan Carter* by The Astronaut and *Get Even* by The Farm 51.

1.1 Goal of the thesis

The objective of the thesis is to help reader using 3D reconstruction to develop realistic model while keeping the representation optimized enough to be used in game engines and video games. 3D reconstruction consists at using captured data from all around the object to create its digital representation. Even if 3D reconstruction is powerful, it unfortunately generates models way too resource costly for live rendering purposes. In this thesis, we will 3D reconstruct assets and search tools to reduce the cost of the representation while keeping most of the details.

1.1.1 Research question

How can we reconstruct real object digitally?

Is it accessible to reconstruct real object digitally?

Can we use 3D reconstructed object in videogames?

1.2 Scope

This thesis will limit itself by using existing tools. The point is to develop a way to achieve acceptable results by finding the solutions that can actually work for videogames. The final product will only be a game prototype with the ability to move and see different assets created with 3D reconstruction.

1.3 Thesis structure

The research will first explain 3D computer graphics and models in different techniques with an emphasis on polygons and point clouds since they are the most used in both videogame and 3D reconstruction technologies. An explanation of textures, baking and UV mapping is also necessary for developing realistic representations. Afterwards, we will present 3D scanning by focusing on laser techniques and the use of photogrammetry. We will finish the theoretical background with game engines and their functions.

The empirical part of the thesis will focus on how to 3D reconstruct video games assets. We will start by comparing the 3D reconstruction technologies and choosing the best for our needs. After that, we will use different programs in our technique to create a high polygon textured mesh. The next step is to create a less detailed model via retopology. This is done for optimization while keeping the details of the reconstructed object by using baking techniques. The last step is to import the baked low poly mesh into Unity Game Engine and see the result of our work.

2 Theoretical framework

2.1 3D Computer graphics

3D Computer graphics is the use of three-dimensional geometric data to represent graphics. The data is then generally calculated into 2D images to be displayed on a screen but it can also be used in interactive computer aided design to visualize the product. (Watt, 2000, p. 27.)

Different techniques for 3D CG exists. Each with advantages and weaknesses. The most commonly used is the Polygonal representation. It consists on approximating an object with a mesh or net of planar polygonal facets. Shading algorithms are then used to create an illusion of high details on more approximate meshes. The disadvantage is that it requires a great amount of polygons and resources to display complex forms and curves. (Watt, 2000, p. 30.)

Bi-cubic parametric patches are curved quadrilaterals. Like the Polygonal representation, the object is created in a mesh. But instead of planar faces, the mesh is composed of curved quadrilaterals, each with its own mathematical formula. This let each curves to be smoother than with the Polygonal technique, but to smooth the joints require a lot of work. Modifying the mesh is also harder because of the need to redo the formula and smoothness of the object. (Watt, 2000, p. 31.)

Another technique is Constructive Solid Geometry. CSG is about creating complex form by combining different primitive object like a cube, sphere or cylinder. The CSG method is volumetric. It represents the volume of the object instead of its surface like the Polygonal representation. Since all the base polyhedron are finite in volume, the end result can only be finite as well, assuring the tightness of the object. This technique is mostly used in Computer Aided Design to help create new object to manufacture. (Laidlaw, Trumbore, & Hughes, 1986)

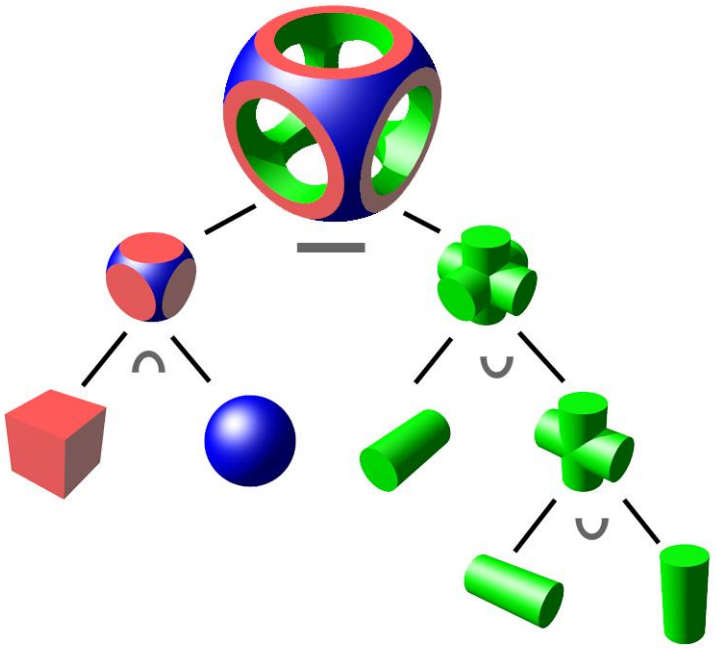

Figure 1 Representation of a CSG tree in POV-Ray (Zottie, 2012)

Objects can also be represented with voxels, 3D pixel. The spatial subdivision technique is about representing the space into cubes. Each cube can be empty or be part of the object. Calculating 3D graphics this way is expensive in resources but it can be used for ray tracing to calculate lights on the object. (Watt, 2000, p. 31.)

We can also represent 3D graphics through point clouds. Generally calculated from laser scanning or photogrammetry, point cloud is just a set of 3D coordinates and values associated with it. They can be transformed into other type and be used as a normal model afterwards. (Gebhardt, et al., 2009)

The last method is the implicit representation. It consists on using the actual mathematic formula for the desired object. Mostly used when interacting with other representation and only for basic polyhedrons. (Watt, 2000, p. 32)

2.2 3D Model

A 3D model is the mathematical representation of a three dimensional object. In case of the polygonal 3D computer graphics technique, the model is composed of vertices, edges and faces. A vertex is a set of three coordinates describing a point in space. Edges are formed by linking two vertices. Faces are generally the surface between three linked edges. Inside the 3D model file, information about the textures and UV coordinates are also stored. (Watt, 2000, p. 34)

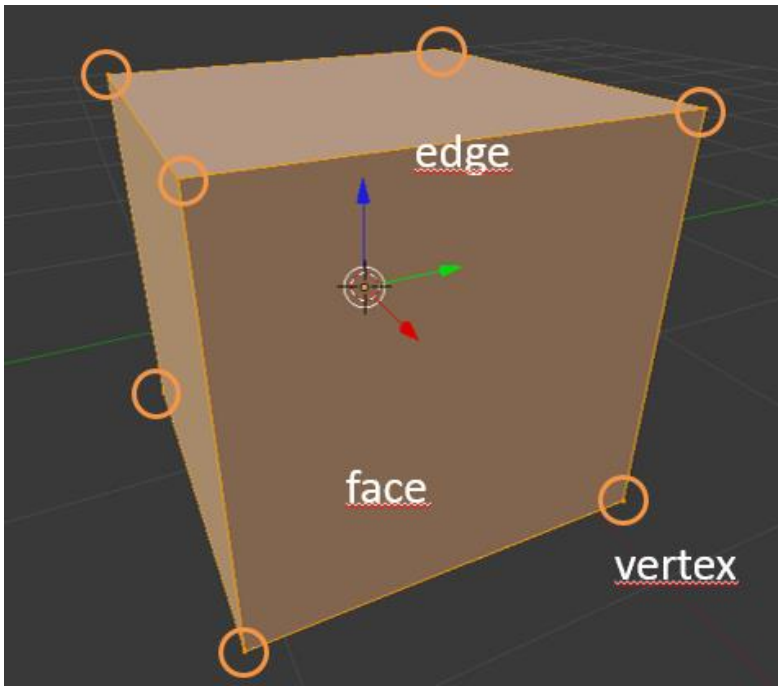

Figure 2 Polygon mesh explanation (Hietala, 2013)

2.2.1 Non-manifold geometry

Non-manifold geometry essentially cannot exist in the real world. They can be unattached edges or vertices, internal faces or areas with no thickness. This can be a problem when trying to 3D print the object, using fluid simulation and rendering refractive effects. (Peels, 2013)

2.2.2 Texture

Texture mapping is the technique in 3D computer graphics consisting in wrapping a 2D image onto a 3D object. This let artist create models with different visual representation without creating multiple object. (Lindquist, 2015) The image can either be seamless and repeat over big surfaces or it can use a unique texture with UV mapping and details every part of the object. (Nietula, 2013)

The modeler can use texture to give models different aspects. The same object can appear smooth, rough, wooden or rocky only by changing the image linked to it. The texture can be any kind of pictures, paintings, illustrations or computer generated images. By the same account, textures can be edited in an image manipulation program. (Luhtapuro, 2014)

2.2.3 UV Mapping

UV mapping comes from the 2D representation of the 3D model. Since the coordinates XYZ are already used for the actual object, the set of UV(W) is used to represent it on the textures and general 2D. The most common technique for UV mapping is to separate the object in large pieces that can then be unfolded. (Ahearn, 2009, pp. 70-72.)

The UV mapping technique can be shown with this cube. The cube unfolds into a 2D plan with 6 squares.

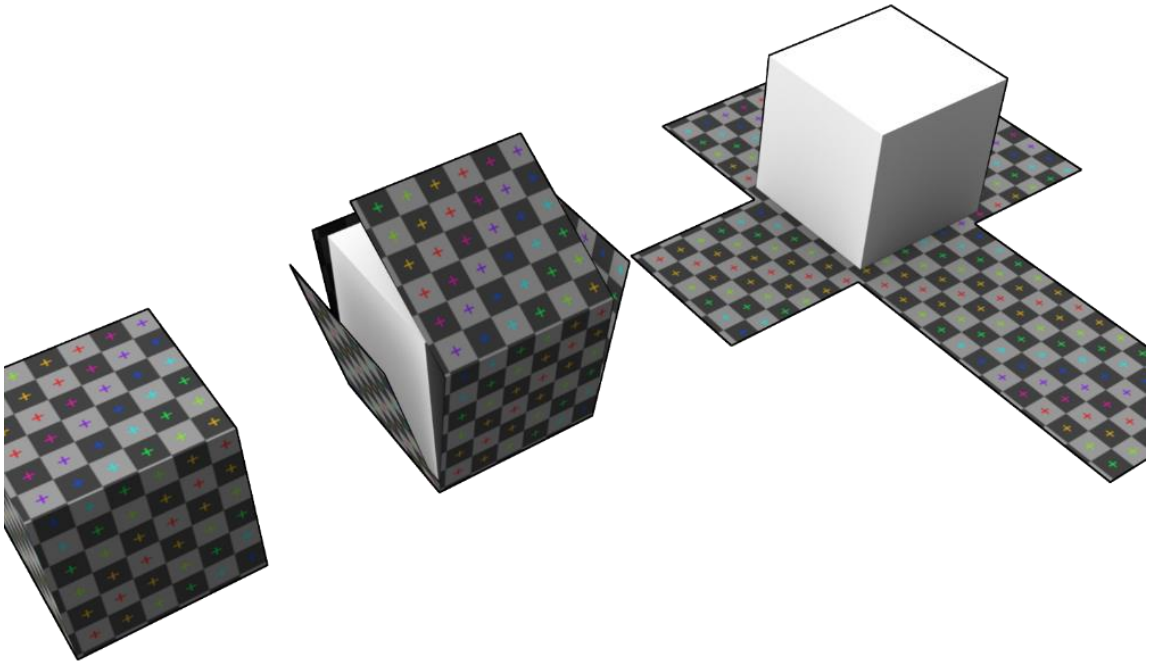

Figure 3 Representation of the UV mapping of a cube. (Zephyris, 2008)

2.2.4 Bake

For IT, baking generally means pre-generating an often reused part of the final process. This is to save time as unbaked object would be otherwise computed multiple times, losing resources. (Blender Foundation, 2013)

Baking in 3D technologies means rendering part of your scene directly on the texture files. This means it can be used to optimize different high cost calculation. For videogames, lights were generally baked onto texture to help with the framerate. (Masters, 2012)

Another use of bakes are the normal maps. These generates illusions of details in models. Normal maps are created by baking a high poly object onto a low poly model. This process lets the low poly asset keeps the details of the less optimized one.

2.3 Point Cloud

Point clouds are typically raw input points from laser scanner or photogrammetry. Each vertex can include a set of 3D coordinates, color and normal direction. The normal direction is important for rebuilding the mesh as it indicates the inside of the object and approximate the surfaces. (Huang, Li, Zhang, Ascher, & Cohen-Or, 2009)

At the start of 3D reconstruction technology, only industrial laser scanner could get the necessary information to create well-made point cloud. Stanford University still offer freely a collection of point cloud from their industrial scanner. They can be used to test new meshing algorithms. (Hyvärinen, 2012)

Now, point clouds can be captured with consumer grade equipment. Camera, Microsoft Kinect or basic laser can acquire information accurately enough to create decent point clouds. (Hyvärinen, 2012)

2.4 3D Reconstruction

Recently, 3D reconstruction has become a popular way to create complex 3D models. The development of different type of 3D shape acquisition can help explain it. Especially the inexpensive optical scanners. However, the generation of a 3D representation still requires long time investment and work. Modeler still needs to clean up manually the data, fill the hole, align and extract the model. (Pauly, Mitra, Giesen, Gross, & Guibas, 2005, p. 2.)

3D scanners are also used in museum to preserve different object, sculptures or even paintings. The highly precise measurements can recreate an accurate record of the target texture and form. As for photography, this only represents the surface of the object. However, we can link 3D data with x-radiography information to recreate high resolution of the internal structure. (Wachowiak & Karas, 2009)

2.4.1 Laser Scanning

Laser scanning is an active 3D scanning technique. As its name suggest, a laser is emitted on the object to get the data. Each emission creates a vertex of the point cloud. There is different type of laser scanning depending on how to receive the reflection of the laser and calculate the position of the object. (Wachowiak & Karas, 2009, p. 148.)
The time-of-flight laser scanner calculates the time between emission and reception of a pulse of light to know the distance of the subject. Since we know the speed of light and the time of the round-trip, the distance will be $c * t / 2$. The accuracy of our scanner depends on the accuracy of our time calculation. (Wachowiak & Karas, 2009, p. 149.)

Miller (2001) explains the properties of a laser triangulation system. It combines a laser with optics to emit onto a target surface. The light will scatter on hit with the subject and will be collected by the optic sensor. The sensor focuses the image onto the position-sensitive detector. This provides the system with two separate current outputs. The system can measure the distance between the sensor and the surface as well as the normal of the point.

2.4.2 Photogrammetry

As stated by Heaven (2014), "Photogrammetry works by gathering hundreds of photographs of an object or scene taken from multiple angles and combining them into a 3D model." An algorithm then finds the common points between photos to create a point cloud of the target. Finally, the vertices of the point cloud can join in a mesh and create a highly detailed polygonal model. (Heaven, 2014)

The result is the exact shape, the exact look and many photos. The software generally also recreates a texture from the data by projecting pixels from photograph to the model. Poznanski (2014) emphasizes that photogrammetry only works with static scene and multiple cameras are necessary for reconstructing humans and animals. For large-scale project, photos can cover only parts of the object. (Poznanski, 2014)

This cliff shows a 242 photos photogrammetry done during the development of the videogame ‘The Vanishing of Evan Carter’

Figure 4 Cliff from 'The Vanishing of Evan Carter' in 3D (Poznanski, 2014)

Poznanski (2014) further points out that photogrammetry is not always simply taking pictures and feeding a software. He argues that the technique is easy to learn but hard to master. To get a clear and clean scan, the photos needs to be net and with no high contrast. Since everything should be static, this includes background and lighting. This can be hard to generate while shooting pictures in the nature. (Poznanski, 2014)

Afterwards the pictures often need to be reworked. Masking parts and processing them before the photogrammetry. Sometimes the software gets confused and needs help to reorient and align the photos. It matters to improve the texture to clear overshadowing and excessive lightning. In addition, the result model will weigh between 2 and 20 million triangles, which is generally the budget for the whole video game or movie (Poznanski, 2014)

2.5 Game Engine

Middleware of videogame creation, a game engine is a collection of tools to create video games. It generally proposes visual development tools and reusable components. The use of a game engine helps developer and artist to focus mostly on the creation of the game. The engine, or framework, generally deals with common tasks like rendering, collision detection, animation, physics, dialog and controls. In addition, game engines are useful to create cross platform games since they can manage dependencies. (Neupane, 2015)

Different game engines exist some are proprietary and reserved to the company using them like Frostbite for EA DICE or Creative Engines for Bethesda Softworks. Others are created especially to be used outside of the developing company like Unity3D, Unreal and CryEngine. (Neupane, 2015)

2.6 Unity3D

Developed by Unity Technologies, Unity3D is a multiplatform game engine. Its development started in 2005. In 2012, the company released Unity version 4 with the Mecanim animation system. This tool helped animator with a comprehensive visual tool for link between animations. It is now popular with developers and is the number one engine for small company in the industry. Since Unity5, the motor is even completely usable freely. The pro version is only necessary after 100,000 euros of revenue per year. With Unity, the games' logic is written in C# and JavaScript. (Neupane, 2015)

The last version of Unity5 is available around 25 different platforms. From desktop, VR and console to mobile and SmartTVs, Unity is nowadays runnable on most consumer machines. (Unity Technologies, 2016)

3 Programs used

YawCam

(Lundvall, 2016)

VisualSFM

(Wu, Agarwal, Curless, & Seitz, 2011) (Wu, 2007) (Wu, 2011) (Wu, 2013)

CMVS PMVS

(Furukawa, Curless , Seitz , & Szeliski, 2010) (Furukawa & Ponce, 2010)

MeshLab

(Cignoni & Ranzuglia, 2015)

Memento Beta

(Autodesk, 2016)

Blender

(Blender Foundation, 2016)

Instant Meshes

(Jakob, Tarini, Panozzo, & Sorkine-Hornung, 2015)

SketchRetopo

(Takayama, Panozzo, Sorkine-Hornung, & Sorkine-Hornung, 2013)

Unity3D

(Unity Technologies, 2015)

4 3D reconstruction

To create a realistic object in a 3D game engine, we will reconstruct a real object digitally. For that, we can either use active 3D scanner like laser technologies or passive reconstruction like photogrammetry. When we compare the two techniques, the main differences for our problem of game development are the accessibility and texture of high definition.

Laser scanners are generally faster and easier to use since they are mostly automated and show direct result. Unfortunately, they require specific hardware that need to be upgraded to keep up with the technology. Lasers can also have problems with highly textured surface. (Leonova, 2014)

On the other hand, 3D reconstruction with photogrammetry needs understanding from the operator and require a great amount of computer calculation. But photogrammetry only needs a camera and calculation power, making it highly accessible. Plus, most of the innovation are software based, so today's captures could be improved with tomorrow's technologies. In addition, photogrammetry creates high definition textures (Leonova, 2014)

For this project, we will use photogrammetry based 3D reconstruction. Mostly because of the great availability of the technology and the need of detailed texture in videogame.

4.1 What will we do

To reconstruct digital model with photogrammetry, we need first to take multiple specific pictures of the object. For the first asset, we will use VisualSFM to create a sparse and a dense point cloud of the object from the pictures. Afterwards we will have to clean the dense cloud from artefacts. For this, we will use MeshLab.

At this point, we can use MeshLab to recreate a mesh from the point cloud. After verification of the mesh's quality, the photos are compiled to develop the object's

texture. We now have a high poly and highly detailed textured model. This result is usable for 3D printing and other application. However, it needs optimization for games and other live rendering use. The optimization will be done in part 6 Export to game engine.

The second asset will be the same, expect we will be using Memento Beta instead of VisualSFM for the photogrammetry. Finally, a comparison of the two models will decide which one will get optimized and used for the rest of the research.

The chosen asset for the project is a stone statue of a knight. This object has enough detail for photogrammetry to be useful. It is also composed of opaque material, ideal for 3D reconstruction. Using the same object for different solutions will let us compare the result.

Figure 5 Statue to be reconstructed in this project

4.2 Pictures

With photogrammetry, the base photos are a crucial part of the process. Actually, if the algorithm fails, it is generally due to poor or confusing set of pictures. (Blizard, 2014) In this part, we will see how to take photographs for photogrammetry.

Ideally, photogrammetry software wants clean, sharp and evenly lit images. Every surfaces of the object need to be represented on multiple angles. If possible, the background should be clear, either with a green wall, a white box or just editing the pictures. This is to take decent photogrammetry images.

We will now look into bad example to assure comprehension on what to avoid. (Sketchfab , 2015)

Table 1 Things to avoid with photogrammetry

Losing focus and movement. Blurry pictures	Blur will confuse the software and create either bad model with artefacts or multiple incomplete representation. The blur is often formed because of a loss of focus or movement in the scene. (Blizard, 2014)
Transparent, shiny or very thin object	Photogrammetry considers object as opaque and non-reflective, plus thin object may not have enough point to create a mesh. As such, the computer cannot understand mirror or glass. The visual distortion created by reflective material will disorient the program and result in a useless deformed model. (Sketchfab , 2015)
Very Repetitive feature	In case of very repetitive features like picket fence or honeycomb, the algorithm can sometimes jump from one part of the repetition to another. If the recurrent motif is in the background, masking it resolve the problem. (Blizard, 2014)
Depth of field	The depth of field matters because photogrammetry needs sharp images of the subject. The subject should be in focus as much as possible in each picture. This implies that you understand your own lens and chose the most adapted setting. (Mallison, 2013)

Lens Distortion	Lens distortion, like fisheye, decreases the potential of the picture. It depends on the algorithm but distortion may create artefact around the object or confusing the alignment and corrupt the whole model. (Blizard, 2014)
JPG compression	Compressing the images may generate false point for the program. The original photos or a lossless copy improves the result by not creating extra artefact. High definition images also boost the quality of the final texture. (Blizard, 2014)
Changing light, shadows and exposure	As for the movement of the scene, movement of the light may be considered as movement of the camera. This false assumption from the algorithms can create distortion in the model or multiple incomplete representation. A static lighting helps clearing confusion. The program may also not get enough detailed in shadowed part of the object. This can result in holes in the mesh. High or low exposure of the scene can result in details loss. Generating a lesser quality reconstruction. Inconsistent exposure can also create light and dark patches on the final representation. These lightning problems can be resolved by controlling the lights. In the nature, clouds create great usable uniform lightning. (Blizard, 2014)

4.2.1 Taking pictures

Ideally, you should consider all these problems before taking pictures. We will need approximately 30 images around the object to create the model. This number increases if you need more than one photo to cover a side of the object, for example, if it is too big. Do not forget any angles because it can be hard to get the asset with the same lighting or the same position.

An important point is to also understand your software and their needed input. For example, VisualSFM will resize pictures over 3200x3200. Possibly reducing the details

of the pictures. Some photogrammetry algorithm also accepts high framerate videos, making it an easier way to get all the angles of the object.

This is an example of a set of photos for 3D reconstruction photogrammetry that was done for this project.

Figure 6 Photo set of the knight for photogrammetry

4.3 Photogrammetry

We will use VisualSFM, MeshLab and Memento for the photogrammetry part of this project. This will create two models that we will compare. The best will continue for optimization.

4.3.1 VisualSFM

VisualSFM is a tool by Wu (2011) for 3D reconstruction using structure from motion. It can be used freely for personal, non-profit and academic purposes. It provides a visual interface to the photogrammetry process and a live reconstruction visualization.

This program is used by firstly adding pictures to the project. Then we can calculate the matches to align the pictures around the object. This is, in general, the longest and

most intensive part of the reconstruction. The program inspects the images and try to find common features between pairs. With these information, it can start to understand the position of the camera for each picture.

The sparse point cloud reconstruction matters for user checking and assure that the images are correctly aligned.

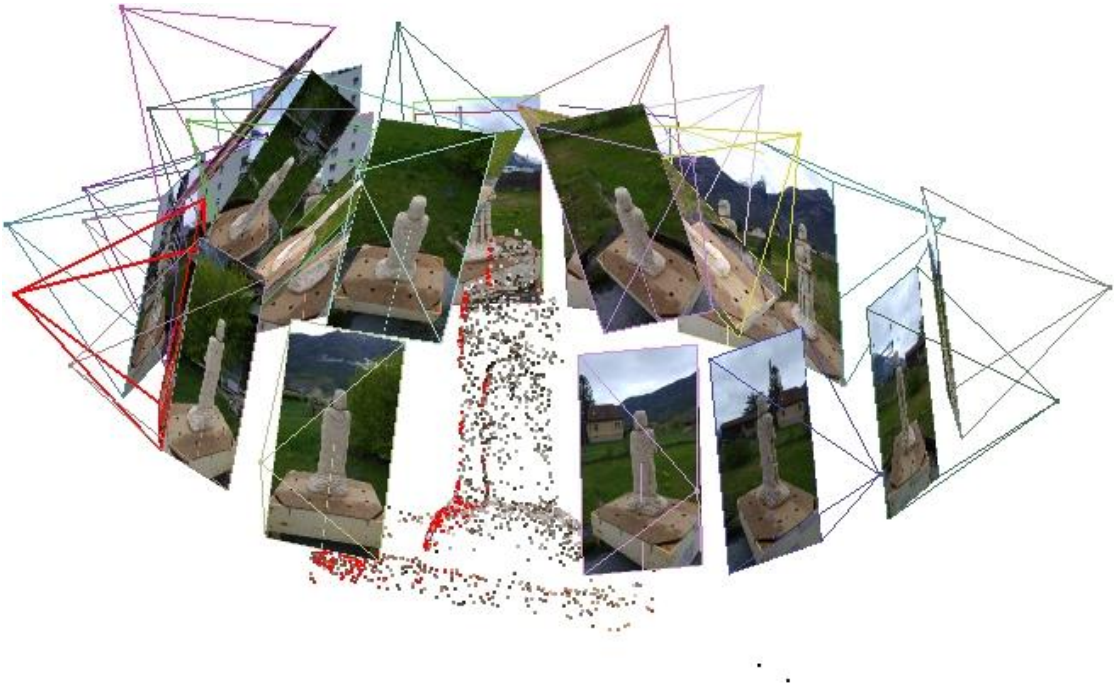

Figure 7 Sparse reconstruction of the knight

After a successful sparse generation, we can launch the dense point cloud reconstruction. This process creates the point cloud that we will use to rebuild the mesh in MeshLab.

4.3.2 MeshLab

MeshLab is an open source tool for processing and editing 3D mesh resulting from 3D reconstruction. We will use this tool to clean and create the mesh from the dense point cloud.

To start using the model, we will need to open the dense cloud's project. For this, open the bundle.rd.out inside the folder created by VisualSFM. This file describes the sparse point cloud. Follow this by opening the list.txt in the same folder. This is the list of

images used during the photogrammetry process, MeshLab will process them to generate the texture.

We can also assure the importation of the data by using the Current Raster Mode of MeshLab. Here, the point cloud follows the image.

Figure 8 Sparse point cloud with raster

Now we can import the actual dense point cloud saved with VisualSFM. The raw dense cloud is generally filled with artefact and other not wanted vertices. With MeshLab, we can clean them out by box selecting then deleting.

Afterwards, we generate the mesh from the point cloud by using the Poisson surface reconstruction algorithm. This part may fail to create a clean mesh. If so, the course of action is to modify the tool parameter until it works as intended.

When the mesh is created and usable, we need to select and delete non-manifold edges since they will trouble the texture generation. The mesh is parametrized and textured from the photos of the base photogrammetry. Equipped with high definition texture, the mesh is now ready to be exported in a retopology tool. For this, we will save the mesh an .obj file, the most universally accepted format in the 3D graphic industry. This export will link the generated texture to the new mesh as well.

4.3.3 Memento Beta

Autodesk proposes freely the Beta version of their new software Memento for test and learning purposes. The solution works by uploading photos of your object to Autodesk's servers. There, they compute the point cloud and generate the mesh with its texture before sending it back to the user. Keep in mind that this solution does not assure privacy of the pictures or the models. This can be negotiated from Autodesk's website.

To start using the Memento solution, you need an Autodesk account. After signing in with your e-mail, Memento proposes to create a model from photos. By following their instruction, you can start the upload of your images.

Example of model creation with Autodesk Memento Beta. The advanced feature Smart Crop serves to automatically crop around a center object. It should be deactivated for complex capture, like urban site from a drone or full interior space. (Vidanom, 2015)

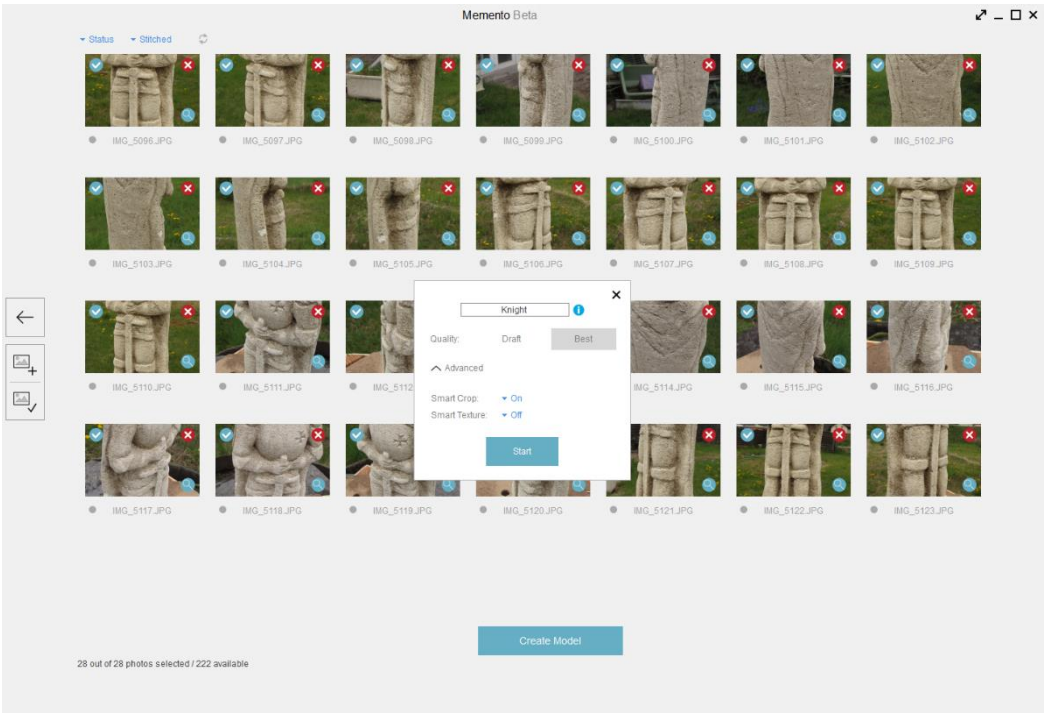

Figure 9 Knight's model creation in Autodesk Memento Beta

The Autodesk's server will then create and texture a model and let you download it back any time you want. The mesh generally comes clean and well-made but it will also

incorporate unwanted part of the photo set, mainly the podium of the object or other foreground element. To arrange that, we need to remove any unwanted faces. We can select and delete using different tools, like box, brush or boundary selection.

Base and clean model comparison from Memento Beta. Removed the box and grass under the knight as well as artefact around the arm.

Figure 10 Comparison between original and cleaned mesh

With the clean model, we can now export the mesh to an obj file. During this process, the texture should be rebaked to a 4096x4096 high definition image. This will allow to retain details which would otherwise disappear. The displacement, normal and ambient occlusion map can also be generated to keep as much feature as possible.

4.3.4 Comparison

Since both software used the same photo set of the same object, we can compare the result to see which seems the best for this application. For this, we will look at the time of creation, manual work required, final structure, texture quality and finally difference in failed projects.

Time

VisualSFM required 5 minutes to create the dense point cloud. 2 minutes was used to clean it. Then MeshLab calculated the mesh and the texture in 6 minutes. Making the whole process at 13 minutes.

With Memento, the project's creation and upload of the pictures took 2 minutes. Followed by 25 minutes of processing server side and 1 minute to download the result. The cleaning and exportation of the mesh was fast with only a minute of work. These result in a 29 minutes' process but was mostly outside work that did not perturb our computer productivity.

Manual work required

For this project, both processes were fairly clean just after the mesh generation. However, VisualSFM and MeshLab still required the user to know how to create the mesh and textures. This process is entirely automatic in Memento.

Final structure

Without the detailed texture, both mesh seems to keep the base form. Though, Memento keeps higher bump details. The Poisson algorithm of MeshLab also created artefacts around both shoulder.

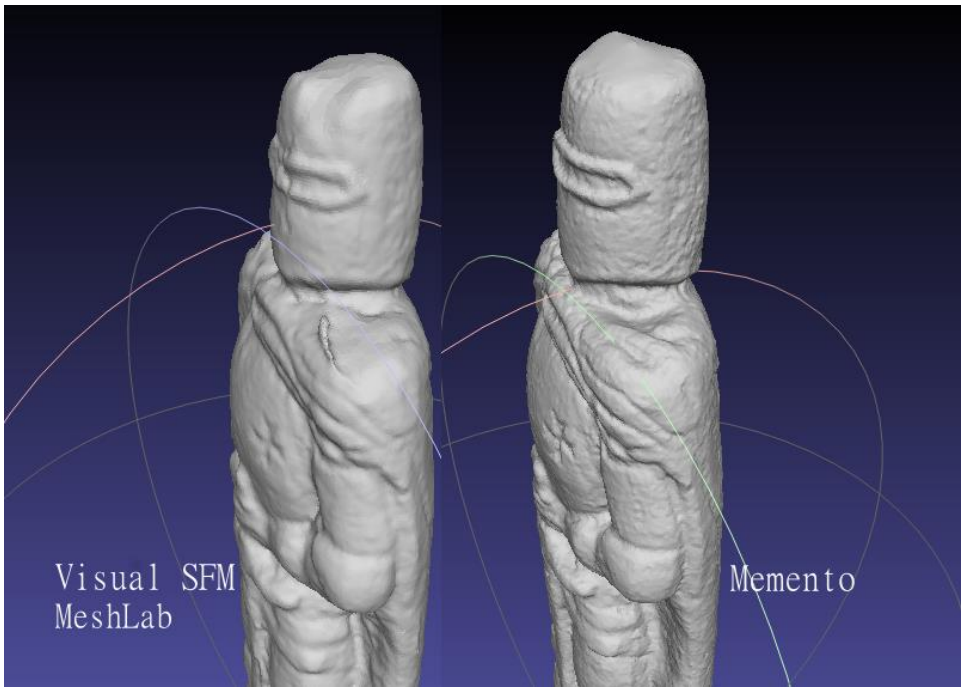

Figure 11 Comparison mesh VisualSFM/MeshLab and Memento

Even with less details, the VisualSFM/MeshLab model actually got more vertices and faces at 281,745 vertices and 562,838 faces to the 207,992 and 415,313 of Memento. The structure from Memento seems cleaner with more detailed and better optimized.

Texture

With the texture, both representation seems correct. Except the upper part of the helm for the Memento version and the left shoulder for VisualSFM/MeshLab.

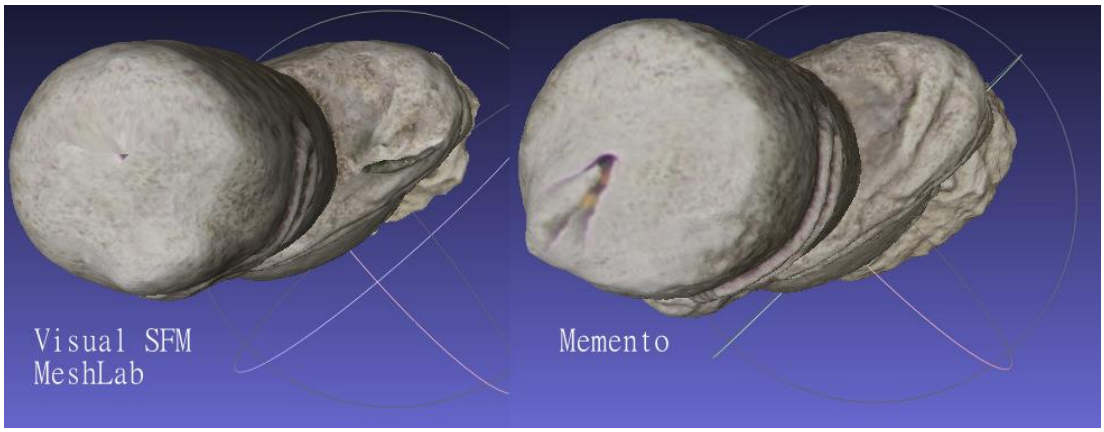

Figure 12 Comparison texture VisualSFM/MeshLab and Memento

Both errors are fairly minor and can be corrected in an image editor. Memento also proposes to create normal, displacement and ambient occlusion map. These may create better looking representation at a minimal cost. Unfortunately, they seem to fail on this particular project.

Failure

It sometimes happens that VisualSFM creates multiple models instead of one with the pictures given. This problem comes from the algorithm not recognizing the object as a whole. Correcting the images in the set or adding some at different angles can resolve it and recreate a whole object.

Failed photogrammetry of a house. VisualSFM generated multiple models instead of aggregating them.

Figure 13 Four dense point clouds of a failed reconstruction in VisualSFM

With Memento Beta, the algorithm seems to try and create a model even if it cannot align the images properly. This behavior results in model corrupted and unusable. On this image, you can see the same picture set of the house with Memento Beta.

Figure 14 Failed mesh reconstruction in Memento Beta

Conclusion

The two assets actually came well-made out of the process. The Memento mesh seems better and require less understanding from the user. This result can however be different from other photo set. In addition, by using Memento, you send your information to Autodesk and cannot control the photogrammetry process. For this thesis, we will continue with the Memento model to create an optimized asset for the Unity game engine.

5 Export to game engine

For using a 3D reconstructed asset in a videogame we need first to optimize it. Since 3D games are generally live rendered, it matters to keep the workload of the video card and processor to a minimum. To optimize the mesh, we will create a new version of it with less faces and vertices. The second way to gain performance on an asset is by reducing its texture's size.

5.1 What will we do

Since we already have a high poly mesh from Part 5 3D reconstruction. We will now optimize it using SketchRetopo and InstantMeshes. Their goal is to build a low poly mesh that follows the base model as much as possible.

Afterwards, we will bake the high poly mesh onto the low poly model with Blender. This generates the same look than the resource costing high poly mesh with the optimized, less detailed representation. Finally, we will use Unity to import the Blender file into a usable asset for our game prototype.

5.2 Retopology

We will create a new mesh around the asset using SketchRetopo and InstantMeshes. This means that we will use the high poly mesh resulting from the photogrammetry as a base to create a low poly optimized model.

5.2.1 SketchRetopo

SketchRetopo is a retopology tool. Like other manual retopology software, the benefit of this program is the ability to keep certain details of the model. But this process takes way more time and practice than automatic retopology. (Takayama, Panozzo, Sorkine-Hornung, & Sorkine-Hornung, 2013)

SketchRetopo takes a bit of time to get used to. See appendix 1 for controls and help on how to use it. Once the tool is learned, it is a fast way to manually change the

topology of your model. The user draws curves on the mesh and the software creates quadrilateral patches. Each boundary can be separated in any number of edges to make low polygon sketching efficient. The patch of quad can be easily modified to follow the curves of the base model.

Different tool, like the laser and cylinder sketching, permits to generate multiple patches at the same time. It also keeps a database of learned patches to use efficient patch on your model.

After some hours of working, all the patch for the knight's project were created then saved in obj format.

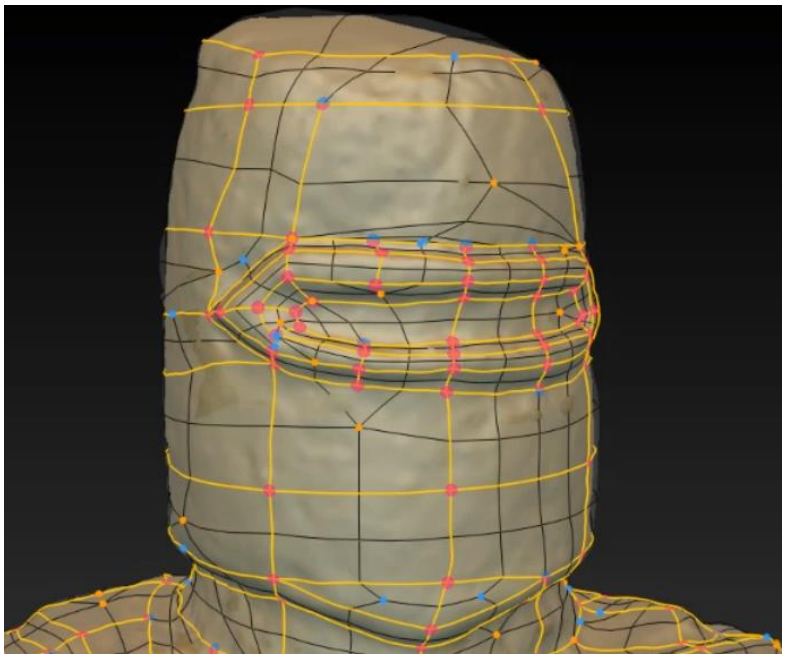

Figure 15 Knight retopology in SketchRetopo

We will now try the same process with InstantMeshes, an automatic retopology tool.

5.2.2 InstantMeshes

InstantMeshes is an automatic retopology software. As stated by Jakob & al. (2015), "We present a novel approach to remesh a surface into an isotropic triangular or quad-dominant mesh using a unified local smoothing operator that optimizes both the edge orientations and vertex position in the output mesh."

This tool can simplify huge amount of polygons into a low polygon version automatically. The user chooses the base model and the amount of vertices they want to conserve. Afterwards, the algorithm generates the orientation of the base model, how its faces seem to flow along the object. This is followed by the actual computation of the different faces around the curve of the model.

The user can comb the orientation of the mesh in more UV-friendly way. Mostly to keep faces logically align for the user. In addition, they have the possibility to force some edges to the program.

For the knight, we can select our target of 13,000 vertices and let the software works for the first step of the process.

Figure 16 InstantMeshes process on the knight

The automatic version of the low poly seems fine but the cape and the stone podium were not aligned. A simple comb brush arranged it. After solving the orientation and position again, the low poly can be exported into an obj file.

This automatic solution for retopology saved time compared to the manual version. What was made in hours after actually learning SketchRetopo, was done in 3 minutes in InstantMeshes.

Since we have now 2 .obj file, we can now compare the result in Blender.

5.3 Baking in Blender

We will now bake the texture and normal of the photogrammetry model onto the low polys that we created. For that we will use Blender. The first step consists in importing both high and low polys in the project. Often the models will import as huge representation in Blender. You can resize them so they are visible and easily usable.

After that we will create a basic UV map. Select your low poly model and enter edit mode. Select all vertices and press U and chose Smart UV Project. Next, in UV Image/Editor add and save a new image, it will be the texture of the model. Go back in Object Mode and find the Bake group under the Render tab of the properties. There we need to change the bake mode to Texture and check Selected to Active. The last part of the baking is to select the high poly then the low one following by pressing the bake button. Create a new image and select the bake mode for the normal.

After the bake of the two bitmap, switch Blender into Cycles Render and add a new Node material to your low poly mesh. In the node editor, you can link the texture, normal and their coordinate to the diffuse. Your optimized model is now ready. Delete the high poly from the project and save.

The three knights in Blender after baking of both the textures and the normal. The result is convincing for both SketchRetopo and InstantMeshes.

Figure 17 The three knights in Blender

5.4 Export to Unity

To use our asset in Unity3D we will create two new assets by importing both blend files. After that, drag & drop the textures and normal into Unity3D. To use the normal map, we need to tell Unity to consider it as a normal. For that, select the normal and change the Texture Type to Normal in the Inspector

Now we need material for the meshes. This is done by dragging the wanted material to the model. Afterwards, the texture and normal can be modified inside the inspector.

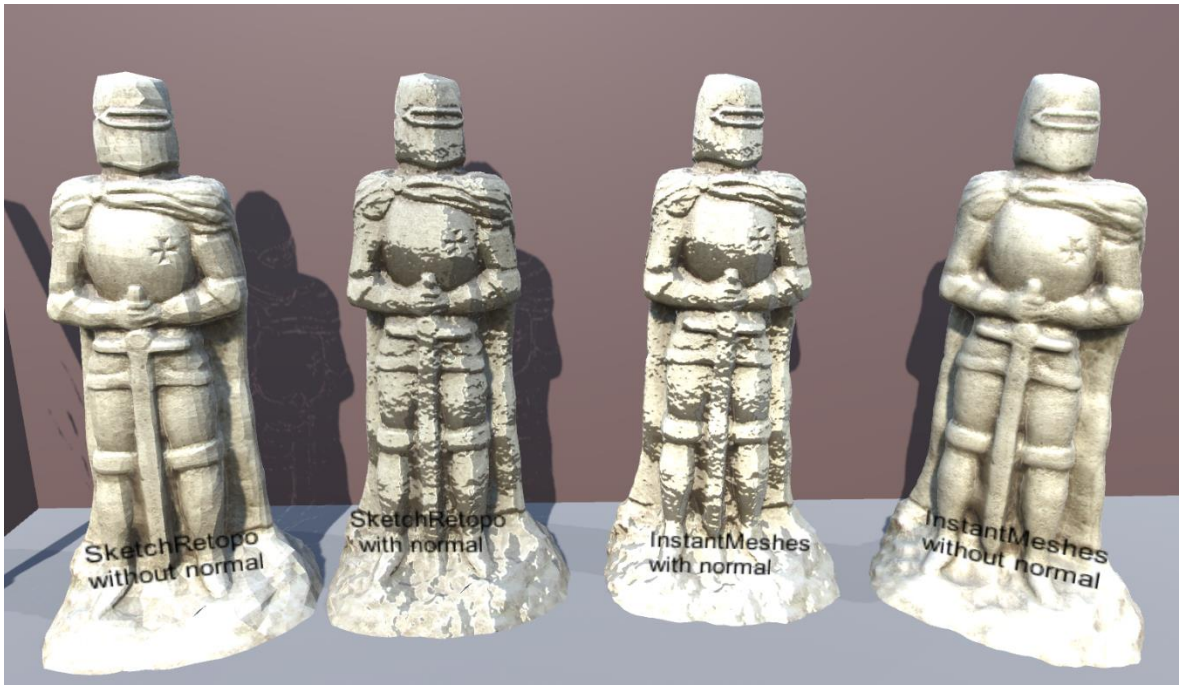

Figure 18 The four knights in a Unity game.

5.5 Comparison

Both models are usable and optimized but the lack of 3D skill in the manual work for SketchRetopo can unfortunately be seen. Even if it is more optimized with only 4986 vertices to the 12685 of InstantMeshes, the loss of quality makes it less useful.

The normal of both mesh seems a bit too hard and create very dark spots. It may be improved with the lightning and by reducing the normal application force in Unity.

The biggest advantage of InstantMeshes is the time-saving of the automatic solution. Since the low poly is generated fairly quickly and without knowledge burden, it can be a valid solution for most purposes. Even if the designer is skilled in retopology, it may be worth their time to try InstantMeshes. The speed of generation makes it quite hard to lose anything precious but can alleviate a huge amount of work.

6 Evaluation

6.1 Result

The project resulted in a videogame prototype in Unity. It is composed of 2 different models, reconstructed from a knight with Memento and optimized in either SketchRetopo or InstantMeshes. Another representation was created with VisualSFM and MeshLab but was not retopologized. We can consider the prototype result a success. Despite this, the models may require the work of a more talented 3D artist to use the full potential of the reconstruction. Indeed, the low poly version seems a bit rough and could use more polish.

The goal of the thesis, explaining how to use reconstruction technology in videogames, was achieved in part 4 and 5. There, the reader is guided on how to digitalize an object and transform it into a videogame asset. The process could actually continue with the art direction of the work. However, this step varies completely depending on the chosen style and is not in the scope of the thesis. Plus, the final asset from this guide actually works like any artist created 3D model. Following standards procedure for the picked design should result in the wanted representation.

I want to reiterate that the result of the choice between the Memento Beta and VisualSFM/MeshLab model was only for this reconstruction. Both software generally gives good result and may fail with different kind of photoset. Since the process is fast, I recommend testing both solution and keep the better output. The process also went remarkably well. The different tools sometimes do not work as cleanly with other set of pictures.

6.2 Sources

Different kind of sources were cited in this thesis.

First, Alan H. Watt is an expert of 3D computer graphics and writer of several books on the subject. The 3rd edition of his general book "*3D Computer Graphics"* was used to

explain the base of the technique. However, even if it covers well the base of 3DCG, this edition dates of 1999 and is obsolete in some section. Mostly the software and limit for optimization changed since that time. The theory behind the different technique, what was used in this thesis, is however still up to date.

Another interesting sources is the article written by Andrzej Poznanski about the use of photogrammetry in the Vanishing of Ethan Carter. He is the art sorcerer of The Astronauts and the sourced article explains how the technique was used. Even if Poznanski is an expert, the article seems to only be personal experience and not a full research on the subject. The source may be less valuable due to lack of peer review and possibility of biased opinion.

Different thesis and research paper were also used for parts of the theoretical background. These were themselves based on research and well-sourced. Each subject being verified with multiple author to remove personal bias.

6.3 Tool selection

The tool selection was based on ease of accessibility. Other techniques exist for each part of the project but were either inaccessible or less performant than the used counterpart. For example, 3D Coat is well-known for retopology and optimization but is not available freely even for educational purposes. Same for Agisoft Photoscan, the most known photogrammetry solution.

The pictures for the project were taken by a Canon PowerShot SX510 HS that I already owned. The quality of the images can change drastically depending on the camera used. Other photos taken by a LG G4 and a Logitech Webcam were also tested before the writing of this project. Both solutions seemed to also work however they generated more artefact due to lack of precision in the settings.

6.4 Learning

During the writing of this thesis, I of course learned how to use 3D reconstruction and photogrammetry. This could be useful in understanding future technology and promote lateral thinking for interface designing. I also increased my knowledge in project management and project writing.

7 Summary

The objective of the thesis was to reconstruct a 3D model from a real object and use it as a videogame asset. The thesis started with gathering information about 3D computer graphic and different reconstruction technologies. In addition, knowledge on how to use 3D asset for videogame and game engines were required.

The goal of the thesis was achieved by using photogrammetry to reconstruct an object that was later optimized though retopology and baking into a usable videogame asset. The thesis also provides information on how to take pictures for photogrammetry and different choice of software for the generation. The research also tackles the use of 3D reconstruction and the benefits for videogames.

Photogrammetry and 3D reconstruction in general helps 3D artist to build highly detailed and realistic 3D model. Many artist uses references and personal knowledge to design their model. This works when the level of detail is low enough to not feel fake realism from the graphics. However, the upgrade in available resources for personal computer drives realist representation in videogame. Photogrammetry helps by giving artist photorealistic usable model for their work.

The programs tested in this thesis all offer great availability and make the process testable by anyone. But they differ in their operation and the result. VisualSFM and MeshLab benefits for the data control and transparence of use. Memento Beta often generates better result with their private algorithm. Then for creating the low poly version of the model, InstantMeshes compute a great timesaving mesh with the problem of having the same size of quad on the whole model. On the other hand, with SketchRetopo the artist can create specific quad patch for the details they want. For the finalization of the asset, Blender was sufficient and the best accessible 3D generic tool.

The thesis result in two functional assets for the game engine Unity3D. The normal of both objects are unfortunately too dark in comparison to the model. This could be solved by tweaking the normal and lightning of the scene.

7.1 Further research/development

The research could continue by trying different technology of reconstruction and means of taking pictures for photogrammetry. For example, laser technology can be used to create precise reconstruction of object without the necessity of learning how to take good pictures. Same for other scanner apparel that can even display the computed model live.

For other kind of photogrammetry, the use of drones could help generate model of building, city and other large scale site. Pixyul, a videogame studio, started a project of capturing Montreal for the in-development game, ReRoll.

Further research could also be done in photogrammetry software. Agisoft Photoscan seems like a great solution and was used during the development of The Vanishing of Evan Carter by The Astronaut. This videogame possesses beautiful realistic asset which were mostly reconstructed with Photoscan.

Another interesting software, the interactive geometry lab of ETHZ proposes an interesting way to use photogrammetry. By using a high framerate video of the object, it manages to separate the foreground from the background. Creating a precise visual hull of the object even for otherwise difficult shapes, like flowers, small trees or small boat.

8 References

Ahearn, L. (2009). *3D Game Textures.* Focal Press.

Autodesk. (2016). Memento Beta. Retrieved from https://memento.autodesk.com

Blender Foundation. (2013). *Render Baking.* Retrieved from Blender Wiki: https://wiki.blender.org/index.php/Doc:2.4/Manual/Render/Bake

Blender Foundation. (2016). Blender 2.77. Retrieved from www.blender.org

Blizard, B. (2014, February 19). *The Art of Photogrammetry: How To Take Your Photos.* Retrieved from http://www.tested.com/art/makers/460142-art-photogrammetry-how-take-your-photos/

Cignoni, P., & Ranzuglia, G. (2015). Meshlab. Retrieved from http://meshlab.sourceforge.net/

Furukawa, Y., & Ponce, J. (2010). Accurate, Dense, and Robust Multi-View Stereopsis. *IEEE Trans. on Pattern Analysis and Machine Intelligence 32 8*, 1362-1376. Retrieved from http://www.di.ens.fr/cmvs/

Furukawa, Y., Curless , B., Seitz , S., & Szeliski, R. (2010). Towards Internet-scale Multi-view Stereo. *CVPR.* Retrieved from http://www.di.ens.fr/cmvs/

Gebhardt, S., Payzer, E., Salemann, L., Fettinger, A., Rotenberg, E., & Seher, C. (2009). *Polygons, Point-Clouds, and Voxels, a Comparison of High-Fidelity Terrain.* Retrieved from www.sisostds.org: https://www.sisostds.org/DesktopModules/Bring2mind/DMX/Download.aspx?Command=Core_Download&EntryId=28745&PortalId=0&TabId=105

Heaven, D. (2014). Picture perfect pixels. *New Scientist 223*, 18.

Hietala, H. (2013, 12 9). Blender courseware. *3*, 6. Retrieved from http://hhmoodle.haaga-helia.fi/mod/resource/view.php?id=374100

Huang, H., Li, D., Zhang, H., Ascher, U., & Cohen-Or, D. (2009). *Consolidation of unorganized point clouds for surface reconstruction.*

Hyvärinen, J. (2012). *Surface Reconstruction of Point Clouds Captured with Microsoft Kinect.* Retrieved from http://www.theseus.fi/handle/10024/42161

Jakob, W., Tarini, M., Panozzo, D., & Sorkine-Hornung, O. (2015). *Instant Field-Aligned Meshes.* ACM SIGGRAPH ASIA 2015. Retrieved from http://igl.ethz.ch/projects/instant-meshes/

Laidlaw, D. H., Trumbore, W. B., & Hughes, J. F. (1986). *Constructive Solid Geometry for Polyhedral Objects* . Retrieved 04 05, 2016, from http://cs.brown.edu/~: http://cs.brown.edu/~jfh/papers/Laidlaw-CSG-1986/main.htm

Leonova, M. (2014, 11 7). *Laser Scanning vs. Photogrammetry*. Retrieved 4 111, 2016, from http://lanmarservices.com/2014/11/07/laser-scanning-vs-photogrammetry/

Lindquist, B. (2015). *Game Environment Texturing Texture Blending and Other Texturing Techniques.*

Luhtapuro, T. (2014). *Surface Detail Mapping in 3D Modelling.* Retrieved from http://www.theseus.fi/handle/10024/73581

Lundvall, M. (2016, 01 31). YawCam. *YawCam 0.5.0*. Retrieved from http://www.yawcam.com

Mallison, H. (2013, November 16). *Photogrammetry tutorial 2: picture taking, general remarks*. Retrieved April 15, 2016, from https://dinosaurpalaeo.wordpress.com/2013/11/16/photogrammetry-tutorial-2-picture-taking-general-remarks/

Masters, M. (2012). *Essential 3D Texturing Terms You Need to Know*. Retrieved from DigitalTutors: http://blog.digitaltutors.com/cover-bases-common-3d-texturing-terminology/

Neupane, P. (2015). *Essential elements of game development : a case study EvilHuman Game.* Retrieved from http://www.theseus.fi/handle/10024/97038

Nietula, E. (2013). *From Vision to Reality: Making of a 3D Environment.* Retrieved from http://www.theseus.fi/handle/10024/56237

Pauly, M., Mitra, N. J., Giesen, J., Gross, M. H., & Guibas, L. J. (2005). Example-based 3D scan completion. In M. Pauly, N. J. Mitra, J. Giesen, M. H. Gross, & L. J. Guibas, *Symposium on Geometry Processing* (pp. 23-32).

Peels, J. (2013). *Why can't 3D printers print non manifold and inverted normal designs*. Retrieved from quora.com: https://www.quora.com/Why-cant-3D-printers-print-non-manifold-and-inverted-normal-designs

Poznanski, A. (2014, 03 25). *Visual revolution of the Vanishing of Ethan Carter.* Retrieved 04 08, 2016, from TheAstronauts: http://www.theastronauts.com/2014/03/visual-revolution-vanishing-ethan-carter/

Sketchfab . (2015). *How to set up a successful photogrammetry project*. Retrieved April 15, 2016, from https://blog.sketchfab.com/how-to-set-up-a-successful-photogrammetry-project/

Takayama, K., Panozzo, D., Sorkine-Hornung, A., & Sorkine-Hornung, O. (2013). *Sketch-Based Generation and Editing of Quad Meshes.* ACM SIGGRAPH 2013.

Unity Technologies. (2015). Unity3D Game Engine.

Unity Technologies. (2016). *Build once Deploy anywhere*. Retrieved 04 10, 2016, from http://unity3d.com/unity/multiplatform

Vidanom. (2015, March 09). *Smart Crop*. Retrieved April 16, 2016, from https://forums.autodesk.com/t5/memento-general-discussion/smart-crop/td-p/5532287

Wachowiak, M., & Karas, B. (2009). *3D scanning and replication for museum and cultural heriate applications.*

Watt, A. (2000). *3D Computer Graphics.* Essex: Pearson Education Limited.

Wu, C. (2007). *SiftGPU: A GPU implementation of Scale Invaraint Feature Transform (SIFT).* Retrieved from http://cs.unc.edu/~ccwu/siftgpu

Wu, C. (2011). *VisualSFM: A Visual Structure from Motion System.* Retrieved from http://ccwu.me/vsfm/

Wu, C. (2013). *Towards Linear-time Incremental Structure From Motion.* 3DV.

Wu, C., Agarwal, S., Curless, B., & Seitz, S. (2011). *Multicore Bundle Adjustment.* CVPR .

Zephyris. (2008). *Cube Representative UV Unwrapping.*

Zottie. (2012, 11 16). *CSG tree.* Retrieved from https://commons.wikimedia.org/w/index.php?curid=263170

APPENDIX A. SketchRetopo Cheat sheet

First get the learned patches database from http://vcg.isti.cnr.it/Publications/2015/MTPPSPC15/ and move it to your user folder with the name *patches.db*

The keyboard layout is assumed to be QWERTY

Basic Command

Ctrl+Z – Undo

Ctrl+Y – Redo

Ctrl+S – Save Retopology to XML

Shift + Right mouse – Diameter of Snapping tool

Camera

Alt + Left mouse – Rotate

Alt + Right mouse – Zoom

Alt + Middle mouse – Drag

Ctrl + Alt + Left mouse – Drag

Q – Basic sketching

Draw curves on the model. Crossing curves create corner of patch. Closed patches create quad polygons.

Ctrl – Continuing an existing stroke

C – Deleting stroke

Shift – Smoothing stroke

B – Create curves around the edge of the object

V – Delete patch

X – Create patch (Hold left mouse from the edge to the patch)

Z – Removes duplicate corners

A – Edit topology

Select a border of a patch to modify the number of edges on the border.
Y – Decrease edges by 1
X – Decrease edges by 2
C – Augment edges by 2
V – Augment edges by 1
Keep in mind that a patch can only create the quad polygons if the sum of edges of the patch is even. Unless for specific reason, it is safer to only modify edges by 2 at a time. Also can move singularities by mouse dragging or with X/C to decrease/increase space between them.

W – Deform curve

Tool to modify a curve. Moving a corner onto another one snap them together
Left mouse dragging – Move part of a curve
Ctrl – Redraw part of a curve

Be careful while using this tool. It often crashes the software and can create weird edges.

T – Edit Corner

Blue circles are not considered a corner of the patch, you can change that by using Edit corner and dragging from the inside of the patch to the desired corner. This can be switched back this the same operation.

Ctrl – Delete corner.

E – Laser sketching

Cut the model with a plane designed with the line dragged by the user.

R – Cylinder sketching

Connect two loops with a cylinder. Needs the two loops to have 0 or the same number of corner.

D – Move vertex

Move vertex, the brush determines the falloff area. Vertices in the brush will move along a bit with the target
This tool is intended as a last touch modification.
Right mouse – Adjust brush size radius
Shift – Smooth the vertices
Ctrl – Compute a temporary UV map to redistribute the vertices more evenly.

F – Edge loop

Can be used to create edge loop at a given place. This will be extended to neighbor patches.
This tool is intended as a last touch modification.

S – Query database

Query the database to find a patch looking like the current one.
Select a patch then draw a curve. The system will look for patches that match the configuration.
Y – Decrease the index
X – Increase the index
Shift – Keep other drawn lines.

APPENDIX B. Unity3D project

VSThesis_Unity3D_Project.zip

APPENDIX C. Photoset

VSThesis_Photoset.zip

APPENDIX D. VisualSFM/MeshLab result

VSThesis_VsfmMeshlab.zip

APPENDIX E. Memento result

VSThesis_Memento.zip

APPENDIX F. Blender result

VSThesis_Blender.bl
end